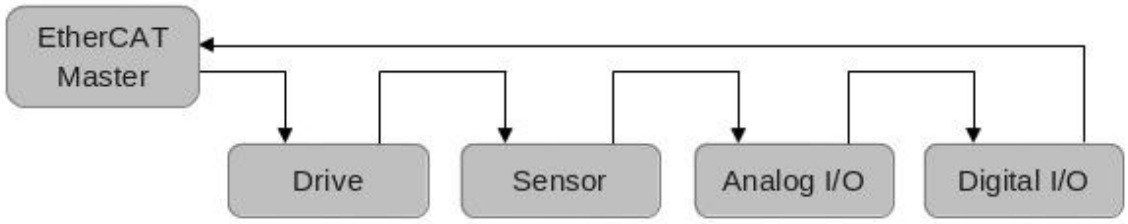
Using the EtherCAT Industrial Communication Protocol In Designs

Maneesh Soni
December 02, 2011

1 of **Enlarge image**

EtherCAT is among the leading communications standards based on Ethernet that is used increasingly for networking and communications in the industrial or factory environment for input/output (I/O) devices, sensors and programmable logic controllers (PLCs). It was invented by Beckhoff Automation in Germany and later standardized by the EtherCAT Technology Group (ETG), established to help with the proliferation of the EtherCAT standard. Today, there are more than 1,700 member companies from 52 countries that create and deploy EtherCAT-compatible products. To ensure broad interoperability among devices designed with EtherCAT interfaces, the EtherCAT Technology Group (ETG) has several rograms for ensuring compliance with the technical specifications.

Ethernet has seen unparalleled adoption in diverse applications, but in an industrial environment, it is still not efficient enough for small amounts of data exchange, it has low determinism for real-time operation, and it works with only star topology in which the network nodes must be connected through switches. EtherCAT technology adds certain features on Ethernet and enforces certain configurations to make it a very efficient network technology for automation while fully conforming to the Ethernet specifications.

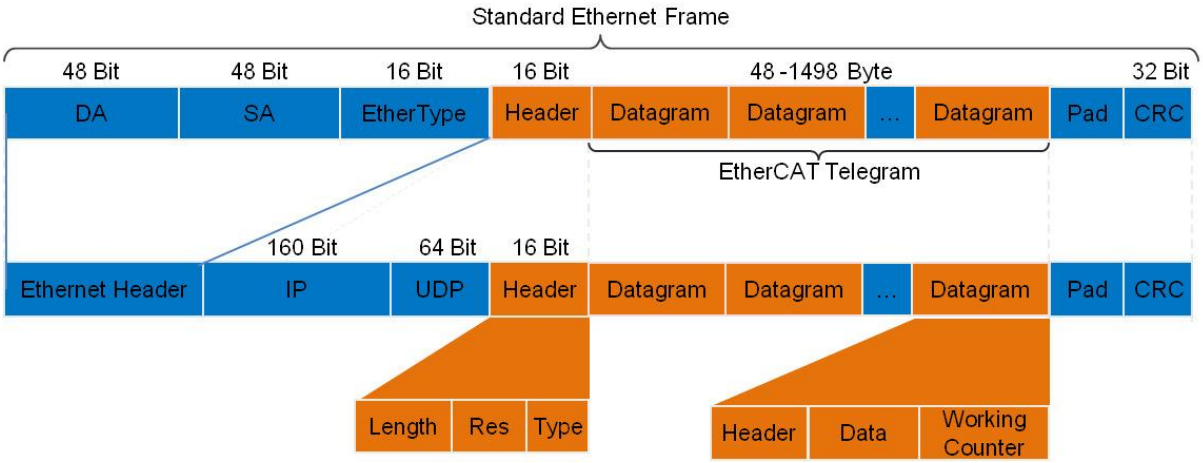


The design of **EtherCAT (Fig. 1)** enables any standard PC to be used as an EtherCAT master and communicate with EtherCAT slaves, which are specialized devices compliant with EtherCAT specification. Together, the master and slave EtherCAT devices can be used in all devices in the factory network – automation controllers, operator interfaces, remote I/O units, sensors, actuators, drives and others. The EtherCAT standard supports any topology — line, star or tree — and the bus structures common in fieldbus networks can also be realized with EtherCAT. Since the EtherCAT interface is present on I/O devices, there is no requirement for any Ethernet switching hardware. With the 100m range of copper links and even longer with optical links, EtherCAT can span over thousands of devices spread across a large geographical area. For short distances, such as on back-plane, EtherCAT uses E-bus, a differential signaling technology.

EtherCAT improves upon traditional Ethernet by implementing “on-the-fly” processing where the nodes in EtherCAT network read the data from a frame as it passes through. All EtherCAT frames originate from the EtherCAT master which sends commands and data to the slaves. Any data to be sent back to master is written by the slave into the frame as it passes through. This helps eliminate the need for point-to-point exchange of small-sized frames between master and individual slaves and drastically improves the efficiency of communication. However, it also means that each slave must have two Ethernet ports and be able to let the frame pass through while reading from or writing to the passing frame and therefore, specialized hardware is required in the slave devices. As a result of these improvements, the usable bandwidth in a

100Mbps network running EtherCAT is more than 90 percent as compared to less than 5 percent for networks where the master must separately communicate with each slave node.

EtherCAT Telegram



The **EtherCAT telegram (Fig. 2)** is encapsulated in Ethernet frame and includes one or more EtherCAT datagrams destined to the EtherCAT slaves. Such Ethernet frames use the EtherCAT type in the header or they can be packed with the IP/UDP header. When the IP header is used, the EtherCAT protocol can also be used across network routers.

Each EtherCAT datagram is a command that consists of a header, data and a working counter. The header and data are used to specify the operation that the slave must perform, and the working counter is updated by the slave to let the master to know that a slave has processed the command.

EtherCAT Protocol

Each slave processes EtherCAT packets “on-the-fly” in that it receives the frame, parses it and takes action if the address specified in an EtherCAT datagram matches its own address, and forwards the entire datagram from its second port while also updating the contents and the CRC of the packet. Through the datagrams, the EtherCAT master addresses the entire address space of up to 4 GB in which up to 65536 EtherCAT slaves, each with 65536 addresses, can be located. The EtherCAT datagrams do not have any restriction on the order in which the slaves are addressed with respect to the actual position of slave nodes in the network.

There are different types of EtherCAT data transmissions – cyclic and acyclic. Cyclic data are the process data that are transferred at periodic intervals or cycle times. Acyclic data is usually non-critical data that can be large in size and usually exchanged in response to a controller command. Some acyclic data, such as diagnostic data, can be critical and have demanding timing requirements. EtherCAT handles these different data transmission requirements through optimized addressing schemes – physical addressing, logical addressing, multiple addressing and broadcast addressing.

To handle various addressing schemes, each slave has a fieldbus memory management unit (FMMU). The FMMU units in each slave enable the EtherCAT protocol to treat various slave devices as part of a 4GB large memory space with slave spaces mapped in it. The EtherCAT master assembles a complete process image during the initialization phase and then makes even bit-level accesses to slave devices via a single EtherCAT command. This capability makes it possible to communicate practically with any number of input/output (I/O) channels across large and small devices spanning the entire fieldbus network via a standard Ethernet controller and standard Ethernet cable.

EtherCAT Performance

As a result of hardware-based FMMU and on-the-fly processing, the EtherCAT network performs at very high levels of efficiency. It enables cycle times of the order of microseconds to communicate from controllers to field devices. The communication efficiency is no longer a bottleneck in industrial networks and brings it in line with the computation speeds of contemporary industrial PCs. For instance, the increased performance makes it possible to run the current loop, in addition to the position loop, for distributed drives over EtherCAT.

EtherCAT Topology

The EtherCAT standard supports any topology — line, star or tree — and the bus structures common in fieldbus networks can also be realized with EtherCAT. Since the EtherCAT interface is present on I/O devices, there is no requirement for any Ethernet switching hardware. With the 100m range of copper links and even longer with optical links, EtherCAT can span over thousands of devices spread across a large geographical area. For short distances, such as on back-plane, EtherCAT uses E-bus, a differential signaling technology.

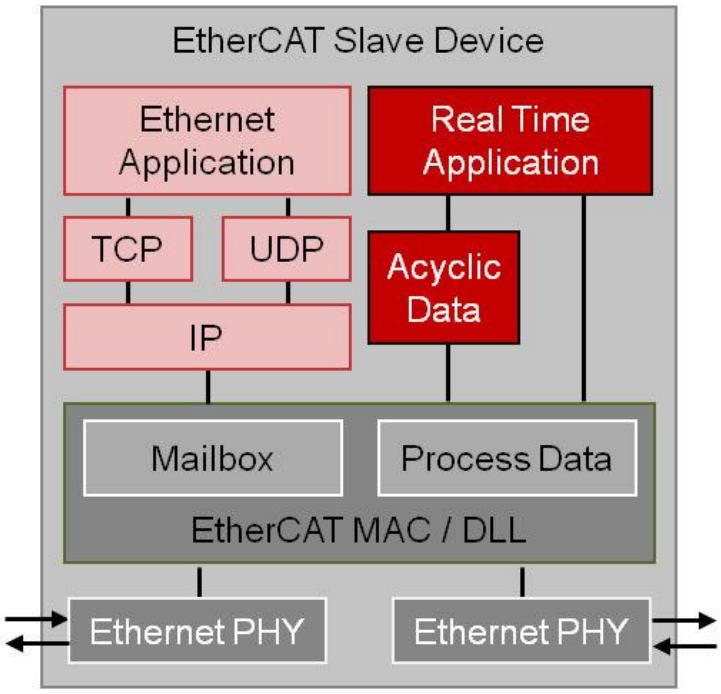
EtherCAT Distributed Clocking

To realize simultaneous actions in industrial nodes installed away from each other, it is necessary to synchronize their internal clocks. EtherCAT accomplishes this by sampling the timestamps for the ingress and egress of an EtherCAT packet on every slave node as it traverses the network. The master uses the timestamp information provided by the slaves to accurately calculate the propagation delay for each individual slave. The clocks in each slave node are adjusted based on this calculation and thus, these clocks are synchronized to within 1µs of each other. An additional advantage of the accurately synchronized clocks is that any measurements taken can be linked to the synchronized time and remove the uncertainty associated with the jitter in the communication between devices.

EtherCAT Device Profiles

In industrial automation, use of device profiles is a very common method to describe the functionality and parameters of the devices. EtherCAT does not provide any new device profiles; instead, it provides interfaces to existing device profiles so that legacy fieldbus devices can be easily upgraded to use EtherCAT. Some of such interfaces are CANOpen over EtherCAT, SERCOS over EtherCAT that enable use of CANOpen and SERCOS by taking advantage of the mapping of their data structures to EtherCAT.

Components of an EtherCAT node

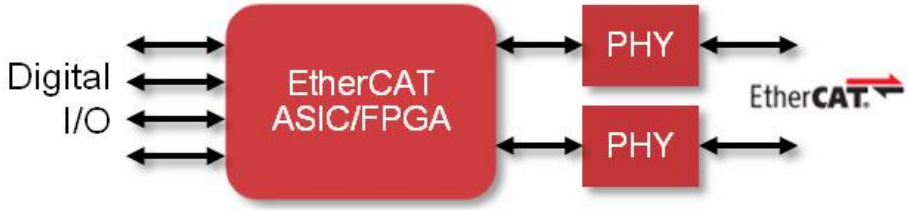


Each EtherCAT node has three components – the physical layer, the data link layer and an application layer **(Fig. 3)**.

The physical layer is implemented using 100BASE-TX copper, 100BASE-FX optical fiber or E-bus based on LVDS signaling. The MAC is implemented either in a specialized ASIC or an FPGA as per the EtherCAT standard specifications. Beyond the MAC is the industrial application that takes care of application specific behavior and a standard TCP/IP and UDP/IP stack to support Ethernet based device profiles. Depending on the complexity of the device, the EtherCAT node can be implemented in hardware or it could be a combination of hardware and software running in an embedded CPU.

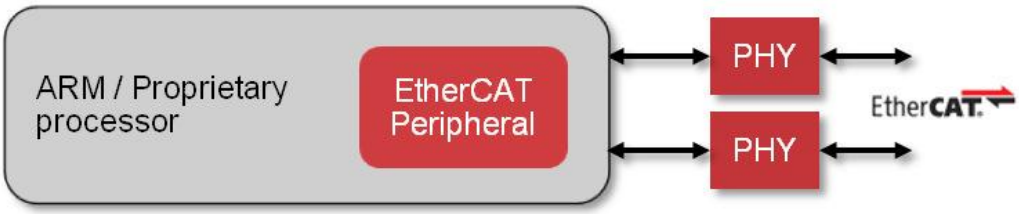
Typical EtherCAT node

A typical EtherCAT node that is in use today has architecture similar to one of the illustrations below.



Many of the simple EtherCAT devices such as digital I/O can be created using single FPGA or ASIC solutions available today. A simplified version of such architecture is shown in **Figure 4**. Such architecture is well suited for cost sensitive simple I/O nodes that do not require software and all functionality is implemented in hardware.

In the EtherCAT nodes where additional processing power is needed, an external processor, often with on-chip flash memory, is connected to the **EtherCAT ASIC/FPGA (Fig. 5)** for handling the application level processing. Such devices could be sensor applications, for instance, where the processor is required to operate the sensor, implement the device driver and run the EtherCAT application stack. The cost of such architecture is higher than that for simple digital I/O devices and it comes with the flexibility that developers can select a processor that suits their needs and cost targets.



In yet another approach, the EtherCAT implementation is one of the peripherals in the device that has an integrated CPU **(Fig. 6)**. Many FPGA devices have the capability to configure a processor in the FPGA or already have an integrated processor. Some vendors provide ASICs with both EtherCAT and a suitable processor on the device. The FPGA are flexible but depending on the CPU selection, there is a risk that cost or operating frequency targets are challenging to meet.

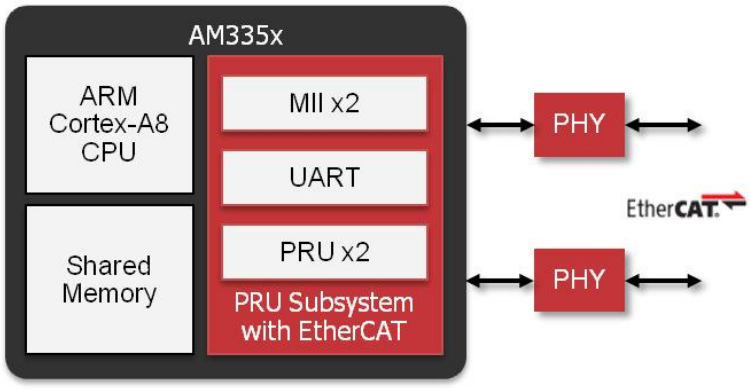
EtherCAT single-chip solution

While the architectures above work perfectly well to implement the EtherCAT protocol, they require two chips, and increase space requirements and bill of materials (BOM) costs for industrial designs. A single-chip solution integrating an ARM Cortex-A8 processor with many other peripherals and interfaces is well-suited for designing industrial automation equipment and can reduce BOM costs by as much as 30 percent.

Integrating a programmable real-time unit (PRU) subsystem, which supports very low level interaction with the MII interfaces, enables the PRU subsystem to implement specialized communication protocols such as EtherCAT. The entire EtherCAT MAC layer is encapsulated in the PRU subsystem through firmware. The PRUs process EtherCAT telegrams on-the-fly, parse them, decode the address, and execute the EtherCAT commands. Interrupts are used for any communication required with the ARM processor where the EtherCAT stack (Layer 7) and the industrial application run. The PRU subsystem also performs the frame forwarding in the reverse direction. Since the PRU subsystem implements all EtherCAT functionality, the ARM processor can be utilized for complex applications or a lower speed variant can be used for simpler and cost-constrained applications, such as distributed I/O.

EtherCAT software architecture

Software is also a very important factor to make sure that the EtherCAT implementation runs smoothly on the devices. There are three major components to consider on a single-chip solution. The first is the micro-code that implements Layer 2 functionality in the PRU; the second is the EtherCAT slave application stack that runs on the ARM MPU and the third is an industrial application that is dependent on the end equipment in which this solution is used.



The recently announced **Sitara AM335x (Fig. 7)** ARM Cortex-A8 microprocessors (MPUs) from **Texas Instruments (TI)** demonstrates this process and illustrates how multiple peripherals can be integrated onto a single ship to implement EtherCAT protocols. In addition to the software listed above, there are additional supporting components, such as the protocol adaptation layer and device drivers that are provided by TI in the software development kit.

With a single-chip solution and industrial software, the industrial automation design possibilities using EtherCAT are infinite.